UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/862,412	05/21/2001	Seth M. Demsey	MS160304.1/40062.100US01	8482

7590 02/02/2006

Homer L. Knearl
Merchant & Gould P.C.
P.O. Box 2903
Minneapolis, MN 55402-0903

EXAMINER	
BULLOCK JR, LEWIS ALEXANDER	
ART UNIT	**PAPER NUMBER**
2195	

DATE MAILED: 02/02/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/862,412	DEMSEY ET AL.	
	Examiner	Art Unit	
	Lewis A. Bullock, Jr.	2195	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 November 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2,4,6-22,24,26,27 and 29-33* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2,4,6-22,24,26,27 and 29-33* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *21 May 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1, 2, 6, 9, 11-20, 24, 26, 27, 29 and 31 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE (U.S. Patent 6,918,106) in view of "Enterprise JavaBeans" by Monson-Haefel (HAEFEL).

As to claim 1, BURRIDGE teaches a computer program product encoding a computer program for executing on a computer system (object oriented computer system), a computer process for generating a customized library (library file / JAR file) for execution of an application (application) by a client (user system), the client including one or more client loaded types already loaded on the client (program units stored in an initial application library file), the computer process comprising: identifying one or more application-referenced types (application program units) on which the application (main program unit) depends for execution but which are not part of the application (via the offline loader / runtime loader determining application program units that are loaded as determined by referenced library file); identifying one or more client needed types (needed application program units) required by the client to execute the application, client needed types comprise one or more application referenced types that are not client loaded types (via the loader determining at a subsequent invocation that the application file is not already loaded in the referenced library file and creating a new

library file having the needed application file); generating the customized library, including the one or more client needed types (loader creating library file containing needed application file that is not loaded); and separately sending the customized library (via loading the file when it is found or when all determined referenced files are placed in the library file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). It would be obvious to one skilled in the art at the time of the invention that the application is sent separate from the file because the application is executed in order to determine which program files to load (see also col. 9, lines 3-13). In addition, Burridge teaches that a main routine is a applet that executes on a browser and specifies the JAR files that it uses wherein JAR files may be stored in multiple places and on multiple servers (col. 2, lines 17-33). Therefore, in order for the applet to use and invoke the library it must be loaded into the local computer and thus Burridge teaches loading the library. Col. 1, line 58-65 details that, "Any java class can be loaded into a running Java interpreter at any time, **possibly even over a network....When the Java application or applet is loaded in this manner,** however each of the files is transferred in uncompressed from using a separate request, making the process relatively inefficient. However, BURRIDGE does not teach explicitly teach that third parties develop the referenced types.

HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to

be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL in order to facilitate package classes to be used by some type of application (pg. 30).

As to claim 24, BURRIDGE teaches a customized library management system (object oriented computer system) for managing a customized class library (library file / JAR file), the system comprising: an application server module (source) receiving an application request and transmitting a requested application to a client (via the application needing an application file that isn't loaded and loading it from a determined path/source); and a customized library generator creating a customized library (library file / JAR file) (via loader creating library file containing needed application file that is not loaded) and separately sending the customized library (library file / JAR file) to the client (via loading the file when it is found or when all determined referenced files are placed in the library file), the customized library including one or more client needed types, which are application referenced types not loaded on the client but required by the application to execute (loader creating library file containing needed application file that is not loaded) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). It would be obvious to one skilled in the art at the time of the invention that the application

is sent separate from the file because the application is executed in order to determine which program files to load (see also col. 9, lines 3-13). However, BURRIDGE does not explicitly teach that third parties develop the referenced types.

HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL in order to facilitate package classes to be used by some type of application (pg. 30).

As to claims 26, BURRIDGE teaches a customized library management system (object oriented computer system) for separately receiving a customized library (library file / JAR file) associated with an application (main program unit / application), the system comprising: a catalog (referenced application library file) identifying one or more client loaded types that are loaded on a client (via program units stored in an initial application library file referenced by the main program unit); a filter module comparing the catalog to a list of one or more application referenced types, which are required by the application to execute (via the loader determining at a subsequent invocation that the application file is not already loaded in the referenced library file and creating a new

library file having the needed application file), and generating a client composite list (another application library file) to identify one or more client needed types, the client needed types comprise one or more of the application-referenced types that are not client-loaded types (loader creating library file containing needed application file that is not loaded); a customized library generator loading the client-needed types from the client composite list into the customized library (via creating a library file and loading the library file for subsequent executions); and a client receipt module for separately receiving the customized library and the application (via loading the file when it is found or when all determined referenced files are placed in the library file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract). It would be obvious to one skilled in the art at the time of the invention that the application is sent separate from the file because the application is executed in order to determine which program files to load (see also col. 9, lines 3-13). However, BURRIDGE does not teach explicitly teach that third parties develop the referenced types.

HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the

teachings of HAEFEL in order to facilitate package classes to be used by some type of application (pg. 30).

As to claim 2, BURRIDGE teaches inserting only client-needed types into the customized library (col. 5, lines 16-46).

As to claim 6, BURRIDGE teaches excluding from the customized library one or more non-identified types, each non-identified type not being referenced by the application (via the developer removing the reference) (col. 7, line 59 – col. 8, line 2).

As to claim 9, BURRIDGE teaches identifying the one or more client-loaded types (program units stored in an initial application library file); and generating a client composite list to identify the one or more client-needed types, the one or more client needed types including the one or more application-referenced types not loaded on the client (via the loader determining at a subsequent invocation that the application file is not already loaded in the referenced library file and creating a new library file having the needed application file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 11, BURRIDGE teaches accessing a client catalog specifying the one or more client loaded types loaded on the client (program units stored in an initial

application library file); and examining the client catalog to identify the one or more client loaded types (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 12, BURRIDGE teaches identifying the one or more client loaded types by receiving a list of the client loaded types from the client (via a loader using a reference to the initial application library file to search for a invoked application file); and evaluating the client loaded types against the application referenced types to identify the client needed types (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 13, BURRIDGE teaches receiving a client composite list specifying the one or more application-referenced types not loaded on the client (via receiving a JAR file of needed classes not loaded on the system as determined by the loader and stored in a library file) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

As to claim 14, BURRIDGE teaches creating a new library (col. 7, lines 49-51); and adding each of the one or more client needed types to the new library to provide the customized library (via creating a new library and adding the needed classes to the library) (col. 5, lines 16-45; col. 5, line 56 – 65; col. 5, line 66 – col. 6, line 10; col. 6, lines 15-28; col. 6, lines 50-52; col. 9, lines 3-13; col. 9, lines 23-29; abstract).

However, BURRIDGE does not teach that the library is empty, i.e. having no types. It would be obvious to one skilled in the art at the time of the invention that since the library is created and then application files are placed in it, that the library file is initially empty and therefore would be obvious in view of the teachings of BURRIDGE that the library file is empty before metadata is added to the file.

As to claim 15, BURRIDGE teaches adding one or more global data fields of each client needed type to the new library (field for storing a class file) (col. 10, lines 19-28).

As to claims 16 and 17, HAEFEL teaches the loading of referenced types (Java classes) in a library (JAR file) for an application wherein third parties develop the referenced types (pg. 30, "JAR files are ZIP files that are used specifically for packaging Java classes that are ready to be used in some type of application."; pg. 90, "As a packaging mechanism, however, the JAR file format is a very convenient way to "shrink-wrap" components and other software for delivery to third parties."). Official Notice is taken in that it is well known in the art that Java classes have methods and data fields and therefore it would be obvious that since referenced types are packaged and they are classes, that the information would include method signatures and code regarding the method for package.

As to claims 18-20, BURRIDGE an application-referenced type or client-loaded type includes a class (col. 8, lines 54-67; col. 9, lines 3-16).

As to claim 27, BURRIDGE teaches a client request module requesting the application from a server (via the loader loading the main program unit) (col. 1, lines 28-29; col. 5, lines 56-59).

As to claim 29, BURRIDGE teaches identifying the application to be optimized (col. 5, lines 56-59). It is obvious to one skilled in the art at the time of the invention that in order to identify the application, an identifier would have to be associated with the application.

As to claim 31, BURRIDGE teaches an install point indicator associated with the application (pathnames for application files) (col. 5, lines 55-65).

3. Claims 4 and 10 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE in view of HAEFEL as applied to claim 1 above, and further in view of "Method to Update Java Class Library in Client Computer at Runtime" by IBM Technical Disclosure.

As to claim 4, the cited combination substantially teaches the claims as detailed above. However, the cited combination does not teach using a version identifier. IBM

teaches identifying a version identifier of loaded classes such that the class loader checks the version of the request class in the client and the server before loading the class (pgs 1-2). Therefore, it would be obvious to combine the teachings of BURRIDGE with HAEFEL and IBM in order to improve the efficiency of the version control of Java classes distributed in the client computers (pg. 1).

As to claim 10, the cited combination substantially teaches the claims as detailed above. However, the cited combination does not teach using a version identifier. IBM teaches identifying a version identifier of loaded classes such that the class loader checks the version of the request class in the client and the server before loading the class such that new versions of requested classes are downloaded as determined (pgs 1-2). Therefore, it would be obvious to combine the teachings of BURRIDGE with HAEFEL and IBM in order to improve the efficiency of the version control of Java clases distributed in the client computers (pg. 1).

4. Claims 7, 8, 30, 32 and 33 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE in view of HAEFEL as applied to claims 1 and 26 above, and further in view of CHAN (U.S. Patent 6,470,494).

As to claims 7 and 8, BURRIDGE and HAEFEL substantially disclose the invention above. However, neither of the references teaches the generating a dependency list. CHAN teaches recursively examining the application to identify one or more type references (classes determined to be needed) associated with an application

referenced type (via upon the execution of an application, a particular class needs to be loaded) (col. 5, lines 65 – col. 6, line 35; and generating a dependency list identifying the one or more application referenced types based on the type references (via analyzing a classes loaded in the jar file to determine if other classes are needed and adding those classes o the input jar file) (col. 7, line 51 – col. 8, line 32). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL and CHAN in order to facilitate the loading of classes without having to specify the location of the classes prior to run-time (col. 3, line 66 – col. 4, line 1).

As to claims 30, 32 and 33, BURRIDGE and HAEFEL substantially disclose the invention above. However, neither of the references teaches the generating a dependency information. CHAN teaches recursively examining the application to identify one or more type references (classes determined to be needed) associated with an application referenced type (via upon the execution of an application, a particular class needs to be loaded) (col. 5, lines 65 – col. 6, line 35; and generating a dependency list identifying the one or more application referenced types based on the type references (via analyzing a classes loaded in the jar file to determine if other classes are needed and adding those classes o the input jar file) (col. 7, line 51 – col. 8, line 32). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL and

CHAN in order to facilitate the loading of classes without having to specify the location of the classes prior to run-time (col. 3, line 66 – col. 4, line 1)

5. Claims 21 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over BURRIDGE in view of HAEFEL as applied to claim 1 above, and further in view of MENACHEMI (U.S. Patent Application Publication 2002/0103810 A1).

As to claims 21 and 22, BURRIDGE and HAEFEL substantially disclose the invention above. However, neither reference teach the use of a device profile. MENACHEMI teaches downloading of a class and a specification of an application, i.e. device profile, by a class loader by creating an empty class and loading the class and its class details in the empty class (pg. 5, paragraphs 0066 – 0070) and receiving a device profile (device profile and specification) specifying a characteristic of the client (pg. 4, lines 0054-0056); and excluding an attribute from the customized library (class), if the attribute is incompatible with the characteristic of the client (via executing part or all of the application based on the device profile and specification received) (pg. 4, lines 0057). Therefore, it would be obvious to one skilled in the art at the time of the invention to combine the teachings of BURRIDGE with the teachings of HAEFEL and MENACHEMI in order to dynamically building at least part of an application at run-time (pg. 2, paragraph 0038).

Response to Arguments

Applicant's arguments filed November 14, 2005 have been fully considered but they are not persuasive. Applicant argued that the combination of Burridge and Haefel does not teach identifying one or more client needed types. The examiner disagrees. As stated by Applicant in the response, a client-needed type is an application-referenced type that is not a client-loaded type. Burridge teaches when the loader encounters a reference to a program unit that has not already been loaded, the loader looks in the library file and if the program unit is not in the library file the loader searches an alternate source for the program unit. This determination by the loader identifies a client-needed type. Applicant statues to determine client-needed types, Burridge must teach something akin to examining the application-referenced types and determining from any client loaded types those application-referenced types not yet loaded on the client which Burridge does not include. The examiner disagrees. Burridge teaches when the loader encounters a reference to a program unit that has not already been loaded, the loader looks in the library file and if the program unit is not in the library file the loader searches an alternate source for the program unit. The library file was previously created by examining a pre-run of the main program and placing all called files of the pre-run, in an overall library file. The called files would constitute application referenced types. Burridge states as indicated above, that if the referenced program unit is not stored in the library the loader searches an alternate source to load the program. Therefore, a client needed type is identified.

Applicant argues that Burridge does not teach generating the customized library, including the one or more client-needed types which are application referenced types that are not client loaded types because the library includes every file needed by the main method. Applicant states that the library taught in Burridge must include client-loaded types, which are explicitly excluded from the customized library claimed in the present invention. The examiner disagrees. The claim at best, teaches that client-needed types are not client-loaded types and that the library includes client needed types. There is no limitation in the claims that detail that the library excludes client loaded types. Applicant is referred to M.P.E.P. 2111 wherein it is stated that claims are given their broadest reasonable interpretation and that reading limitations in the specification or not recited in the claims is improper in interpreting the claim language. In addition, M.P.E.P. 2111 states:

> The transitional term "comprising", which is synonymous with "including," "containing," or "characterized by," is inclusive or open-ended and does not exclude additional, unrecited elements or method steps. See, e.g., > *Mars Inc. v. H.J. Heinz Co.*, 377 F.3d 1369, 1376, 71 USPQ2d 1837, 1843 (Fed. Cir. 2004) ("like the term comprising,' the terms containing' and mixture' are open-ended.")< *Invitrogen Corp. v. Biocrest Mfg., L.P.*, 327 F.3d 1364, 1368, 66 USPQ2d 1631, 1634 (Fed. Cir. 2003) ("The transition comprising' in a method claim indicates that the claim is open-ended and allows for additional steps."); *Genentech, Inc. v. Chiron Corp.*, 112 F.3d 495, 501, 42 USPQ2d 1608, 1613 (Fed. Cir. 1997) ("Comprising" is a term of art used in claim language which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the claim."); *Moleculon Research Corp. v. CBS, Inc.*, 793 F.2d 1261, 229 USPQ 805 (Fed. Cir. 1986); *In re Baxter*, 656 F.2d 679, 686, 210 USPQ 795, 803 (CCPA 1981); *Ex parte Davis*, 80 USPQ 448, 450 (Bd. App. 1948) ("comprising" leaves "the claim open for the inclusion of unspecified ingredients even in major amounts"). >In *Gillette Co. v. Energizer Holdings Inc.*, 405 F.3d 1367, 1371-73, 74 USPQ2d 1586, 1589-91 (Fed. Cir. 2005), the court held that a claim to "a safety razor blade unit comprising a guard, a cap, and a group of first, second, and third blades" encompasses razors with more than three blades because the transitional phrase "comprising" in the preamble and the phrase "group of" are presumptively open-ended. "The word comprising' transitioning from the preamble to the body signals that the entire claim is presumptively open-ended." *Id.* In contrast, the court noted the phrase "group consisting of" is a closed term, which is often used in claim drafting to signal a "Markush group" that is by its nature closed. *Id.* The court also emphasized that reference to "first," "second," and "third" blades in the claim was not used to show a serial or numerical limitation but instead was used to distinguish or identify the various members of the group. *Id.*<

As outlined above, the term "including" is open-ended and **does not exclude additional, unrecited elements or method steps**. Based on this understanding, Applicant's argument that the library also includes the client loaded types are unpersuasive, since the library includes the client needed types and does not preclude the client loaded types from being stored in the library.

Applicant argues that Burridge does not teach separately sending the customized library and the application to the client. Applicant further states that the application is already present on the client for optimization and that the library is stored locally and is never sent. The examiner disagrees. The background of Burridge provides understanding of what is considered a main routine and a library. In addition, Burridge states through the patent that the invention is related to the java technology (col. 4, lines 56-62; col. 5, lines 1-8; col. 11, lines 32-36). In the background, Burridge teaches that a main routine is a applet that executes on a browser and specifies the JAR files that it uses wherein JAR files may be stored in multiple places and on multiple servers (col. 2, lines 17-33). Therefore, in order for the applet to use and invoke the library it must be loaded into the local computer and thus Burridge teaches loading the library. Col. 1, line 58-65 details that, "Any java class can be loaded into a running Java interpreter at any time, **possibly even over a network....When the Java application or applet is loaded in this manner,** however each of the files is transferred in uncompressed from using a separate request, making the process relatively inefficient". Therefore, Burridge alludes to the Java applet being loaded over a network also. It would be inherent to the teaching of Burridge that since the Java applet and library are sent over a network loaded, that such a loading would have to be either user-requested or automatically requested by the system and thus part of the loading operation performed by Burridge.

Applicant argues that there is no motivation to combine Haefel and Burridge because Burridge teaches away from the combination because Burridge relates to a shared library on a single system and not sending the library to a client. The examiner

disagrees. As explained above, Burridge relates to sending a library over a network to be loaded and thus does not teach away from the combination. Therefore, the rejection is maintained.

Conclusion

6. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 a.m. - 5:00 p.m..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

January 27, 2006

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER